Algorithms for Bandit Problems

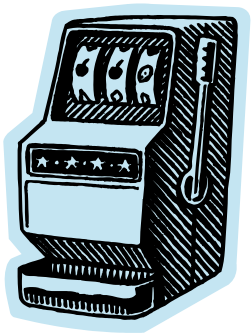
Junya Honda
The University of Tokyo / RIKEN

Special Topics in Mechano-Informatics II
July 4, 2018

Multi-armed Bandit Problems

- Model of a gambler playing slot machines
  (“one-armed bandits”)
- Reward distribution differs between arms
- Hopes to pull the most beneficial arm as much as possible






exploration vs exploitation dilemma:

- Expectations of arms cannot be known without many trials
- Needs to pull the seemingly best arm to earn the reward

Applications

- Since 1930s
  - Clinical trials
  - Choice of crop plants
  - ⋮
- Recent new applications
  - Recommendation of ads and news
  - Network routing
  - Optimization of Search Engines
  - ⋮

Outline

- Classification of bandit problems
- Regret minimization for stochastic bandits
- Algorithms for other formulations

Classification of Bandit Problems

- Mechanism of rewards:
  - Stochastic bandits
  - Adversarial bandits
- Objective of the player:
  - Maximization of cumulative rewards (regret minimization)
  - Identification of the arm with the best mean
- Extensions:
  - Nonstationary rewards
  - Linear / contextual bandits
  - Comparison-based bandits bandits (dueling bandits)

⋮

Stochastic Bandits

- Pulls an arm at each round t
- Reward from the arm i follows Bernoulli distribution Ber(μ_i): 1 with prob. μ_i and 0 with prob. 1- μ_i
- Can observe the reward only for the pulled arm
  (cf. online learning: can observe rewards from all arms)

Stochastic Bandits

- Pulls an arm at each round t
- Reward from the arm i follows Bernoulli distribution Ber(μ_i):
- Can observe the reward only for the pulled arm

Cumulative reward maximization (=regret minimization):

- If the best arm $i^* = \text{argmax}_i \mu_i$ were known:
  optimal to always pull i^* with expectation $\mu^* = \mu_{i^*}$
- Goal: minimize the regret (cumulative loss)

$$\text{regret}(T) = \sum_{i=1}^{K} (\mu^* - \mu_i) N_i(T) = \sum_{i=1}^{K} \Delta_i N_i(T)$$

  - $N_i(t)$: number of times that arm i is pulled through T rounds

Stochastic Bandits

- Pulls an arm at each round *t*
- Reward from the arm *i* follows Bernoulli distribution Ber(μ_i):
- Can observe the reward only for the pulled arm

Best arm identification:

- Fixed confidence: find the best arm through rounds as few as possible with success prob. $\geq 1 - \delta$
- Fixed budget: detect the best arm after *T* rounds with success probability as high as possible
  - Many problems are left open in the fixed budget setting

Adversarial Bandits

- Reward from each arm $x_{i,t} \in [0, 1]$ is determined by an adversary so that the player incurs loss

	t=1	t=2	t=3	t=4
arm 1				
arm 2				
arm 3				

Adversarial Bandits

- Reward from each arm $x_{i,t} \in [0, 1]$ is determined by an adversary so that the player incurs loss

	t=1	t=2	t=3	t=4
arm 1	0.8			
arm 2	0.3			
arm 3	0.4			

Adversarial Bandits

- Reward from each arm $x_{i,t} \in [0, 1]$ is determined by an adversary so that the player incurs loss

	t=1	t=2	t=3	t=4
arm 1	0.8			
arm 2	0.3			
arm 3	0.4			

Adversarial Bandits

- Reward from each arm $x_{i,t} \in [0, 1]$ is determined by an adversary so that the player incurs loss

	t=1	t=2	t=3	t=4
arm 1	0.8	0.5		
arm 2	0.3	0.9		
arm 3	0.4	0.1		

Adversarial Bandits

- Reward from each arm $x_{i,t} \in [0, 1]$ is determined by an adversary so that the player incurs loss

	t=1	t=2	t=3	t=4
arm 1	0.8	0.5		
arm 2	0.3	0.9		
arm 3	0.4	0.1		

Adversarial Bandits

- Reward from each arm $x_{i,t} \in [0, 1]$ is determined by an adversary so that the player incurs loss

	t=1	t=2	t=3	t=4
arm 1	0.8	0.5	0.2	
arm 2	0.3	0.9	0.1	
arm 3	0.4	0.1	0.4	

Adversarial Bandits

- Reward from each arm $x_{i,t} \in [0, 1]$ is determined by an adversary so that the player incurs loss

	t=1	t=2	t=3	t=4
arm 1	0.8	0.5	0.2	
arm 2	0.3	0.9	0.1	
arm 3	0.4	0.1	0.4	

Adversarial Bandits

- Reward from each arm $x_{i,t} \in [0, 1]$ is determined by an adversary so that the player incurs loss

	t=1	t=2	t=3	t=4
arm 1	0.8	0.5	0.2	0.3
arm 2	0.3	0.9	0.1	0.8
arm 3	0.4	0.1	0.4	0.6

Adversarial Bandits

- Reward from each arm $x_{i,t} \in [0, 1]$ is determined by an adversary so that the player incurs loss

	t=1	t=2	t=3	t=4	Total
arm 1	0.8	0.5	0.2	0.3	1.8
arm 2	0.3	0.9	0.1	0.8	2.1
arm 3	0.4	0.1	0.4	0.6	1.5

- regret = maximum cumulative reward of a single arm
  - actual cumulative reward
  = 2.1 - 1.5 = 0.6
- No need to assume some reward distribution

Adversarial Bandits

- Reward from each arm $x_{i,t} \in [0, 1]$ is determined by an adversary so that the player incurs loss

	t=1	t=2	t=3	t=4	Total
arm 1	0.8	0.5	0.2	0.3	1.8
arm 2	0.3	0.9	0.1	0.8	2.1
arm 3	0.4	0.1	0.4	0.6	1.5

- Natural question:
  why doesn't the adversary always set all rewards to zero?
  - The regret is also always zero for all-zero rewards

Adversarial Bandits

- Reward from each arm $x_{i,t} \in [0, 1]$ is determined by an adversary so that the player incurs loss

	t=1	t=2	t=3	t=4	Total
arm 1	0.0	0.0	0.0	0.0	0.0
arm 2	0.0	0.0	0.0	0.0	0.0
arm 3	0.0	0.0	0.0	0.0	0.0

- Natural question:
  why doesn't the adversary always set all rewards to zero?
  - The regret is also always zero for all-zero rewards

Outline

- Classification of bandit problems
- Regret minimization for stochastic bandits
  - ε-greedy
  - UCB algorithm
  - Thompson sampling
- Algorithms for other formulations

Stochastic Bandits

- Pulls an arm at each round t
- Reward from the arm i follows Bernoulli distribution Ber(μ_i):
- Can observe the reward only for the pulled arm

Cumulative reward maximization (=regret minimization):

- If the best arm $i^* = \text{argmax}_i \mu_i$ were known:
  optimal to always pull i^* with expectation $\mu^* = \mu_{i^*}$
- Goal: minimize the regret (cumulative loss)

$$\text{regret}(T) = \sum_{i=1}^{K} (\mu^* - \mu_i) N_i(T) = \sum_{i=1}^{K} \Delta_i N_i(T)$$

  - $N_i(t)$: number of times that arm i is pulled through T rounds

ε-greedy Algorithm

- One of the simplest bandit algorithms
- Partitions T rounds into εT rounds for exploration and $(1-\varepsilon)T$ rounds for exploitation ($0 < \varepsilon < 1$)
- Exploration phase: pulls each arm $N_\varepsilon = \varepsilon T/K$ times
- Exploitation phase: continues to pull the arm that performed the best in the exploration phase
- Regret upper bound:

$$\mathrm{E}[\mathrm{Regret}] \leq \epsilon T + (1-\epsilon) T \sum_{j \neq i^*} \Pr[\hat{\mu}_j \geq \hat{\mu}_{i^*}]$$

  - $\hat{\mu}_i$: empirical mean of arm i after exploration phase
- How should we set ε?

Regret of ε-greedy Algorithm (1/2)

- Misidentification in the exploration phase: $\Pr[\hat{\mu}_j \geq \hat{\mu}_{i^*}]$
- Hoeffding's inequality: for i.i.d. (independently and identically distributed) $Z_1, Z_2, \cdots \in [-1/2,\ 1/2]$ and $\mathrm{E}[Z_i] \geq x$,

$$\Pr\left[\frac{1}{N}\sum_{n=1}^{N} Z_i \leq x\right] \leq \mathrm{e}^{-2N(\mathrm{E}[Z_i]-x)^2}$$

- The misidentification prob. is bounded by

$$\Pr\left[\frac{1}{N_\epsilon}\sum_{n=1}^{N_\epsilon} \frac{X_{i^*,n} - X_{j,n}}{2} \leq 0\right] \leq \mathrm{e}^{-2N_\epsilon((\mu^*-\mu_j)/2)^2}$$

$$= \mathrm{e}^{-N_\epsilon \Delta_i^2/2}$$

Regret of ε-greedy Algorithm (2/2)

- The regret is bounded for $\Delta = \min_{i \neq i^*} \Delta_i$ by

$$\begin{aligned}
\mathrm{E}[\mathrm{Regret}] &\leq \epsilon T + (1-\epsilon) T \sum_{j \neq i^*} \Pr[\hat{\mu}_{i^*} \leq \hat{\mu}_j] \\
&\leq \epsilon T + (1-\epsilon) T \sum_{j \neq i^*} \mathrm{e}^{-N_\epsilon \Delta_i^2 / 2} \\
&\leq \epsilon T + KT\mathrm{e}^{-\epsilon T \Delta^2 / 2K}
\end{aligned}$$

- If ε = $2K(\log T)/T\Delta^2$ then the left term becomes dominant:

$$\mathrm{E}[\mathrm{Regret}] \leq 2K(\log T)/\Delta^2 + K$$

Property of ε-greedy Algorithm

- For $\varepsilon = 2K(\log T)/T\Delta^2$,

$$\mathrm{E}[\mathrm{Regret}] \leq 2K(\log T)/\Delta^2 + K$$

- If ε is set properly then achieves a logarithmic regret
- ε has to be set depending on $\Delta = \min_{i \in \{1,2,\cdots,K\}} \Delta_i$:
- If ε is too small then suffers polynomial regret
  - Impossible to always achieve a logarithmic regret!

Outline

- Classification of bandit problems
- Regret minimization for stochastic bandits
  - ε-greedy
  - UCB algorithm
  - Thompson sampling
- Algorithms for other formulations

UCB Algorithm

- Based on an upper bound of the confidence interval of the mean of each arm (Upper Confidence Bound: UCB)
- Pulls the arm maximizing UCB score at the *t*-th round:

$$\mathrm{UCB}_i = \underbrace{\hat{\mu}_i(t)}_{\text{exploitation}} + \underbrace{\sqrt{\frac{\log t}{2N_i(t)}}}_{\text{exploration}}$$

- Sum of exploitation and exploration terms
- Exploration term corresponds to confidence level $1/t$

Derivation of UCB Score

- Hoeffding's inequality: $\Pr[\hat{\mu}_i \leq x] \leq \mathrm{e}^{-2N_i(t)(\mu_i - x)^2}$
  - Likelihood of the mean parameter μ_i is roughly approximated by $\mathrm{e}^{-2N_i(t)(\mu_i - \hat{\mu}_i)^2}$

- Upper bound of μ_i such that the likelihood is larger than $1/t$:

$$\mathrm{UCB}_i = \max\{\mu_i : \mathrm{e}^{-2N_i(t)(\mu_i - \hat{\mu}_i)^2} \geq 1/t\}$$

$$= \hat{\mu}_i + \sqrt{\frac{\log t}{2N_i(t)}}$$

Regret of UCB Algorithm

- UCB algorithm always achieves a logarithmic regret:

$$\mathrm{E}[\mathrm{Regret}(T)] \leq \sum_{i \neq i^*} \frac{\log T}{2\Delta_i} + o(\log T)$$

(adapted from [Auer+, 2002])

- Lower bound for most algorithms [Lai & Robbins, 1985]:

$$\mathrm{E}[\mathrm{Regret}(T)] \geq \sum_{i \neq i^*} \frac{\Delta_i \log T}{D(\mu_i \| \mu^*)} - o(\log T)$$

  - $D(p\|q) = p \log \frac{p}{q} + (1-q) \log \frac{1-p}{1-q}$: KL divergence

- Bound of UCB is tight in the sense that $D(\mu_i \| \mu^*) \geq 2\Delta_i^2$ and this inequality cannot be improved

Improvement of UCB Algorithm (1/2)

- Chernoff-Hoeffding's inequality: $\Pr[\hat{\mu}_i \leq x] \leq \mathrm{e}^{-N_i(t)D(x\|\mu_i)}$
- UCB score derived from Hoeffding's inequality:

$$\mathrm{UCB}_i = \max\{\mu_i : \mathrm{e}^{-2N_i(t)(\mu_i-\hat{\mu}_i)^2} \geq 1/t\} = \hat{\mu}_i + \sqrt{\frac{\log t}{2N_i(t)}}$$

- UCB score derived from Chernoff-Hoeffding's inequality:

$$\mathrm{UCB}_i = \max\{\mu_i : \mathrm{e}^{-N_i(t)D(\hat{\mu}_i\|\mu_i)} \geq 1/t\}$$

$$= \mu_i > \hat{\mu}_i \text{ such that } D(\hat{\mu}_i\|\mu_i) = \frac{\log t}{N_i(t)}$$

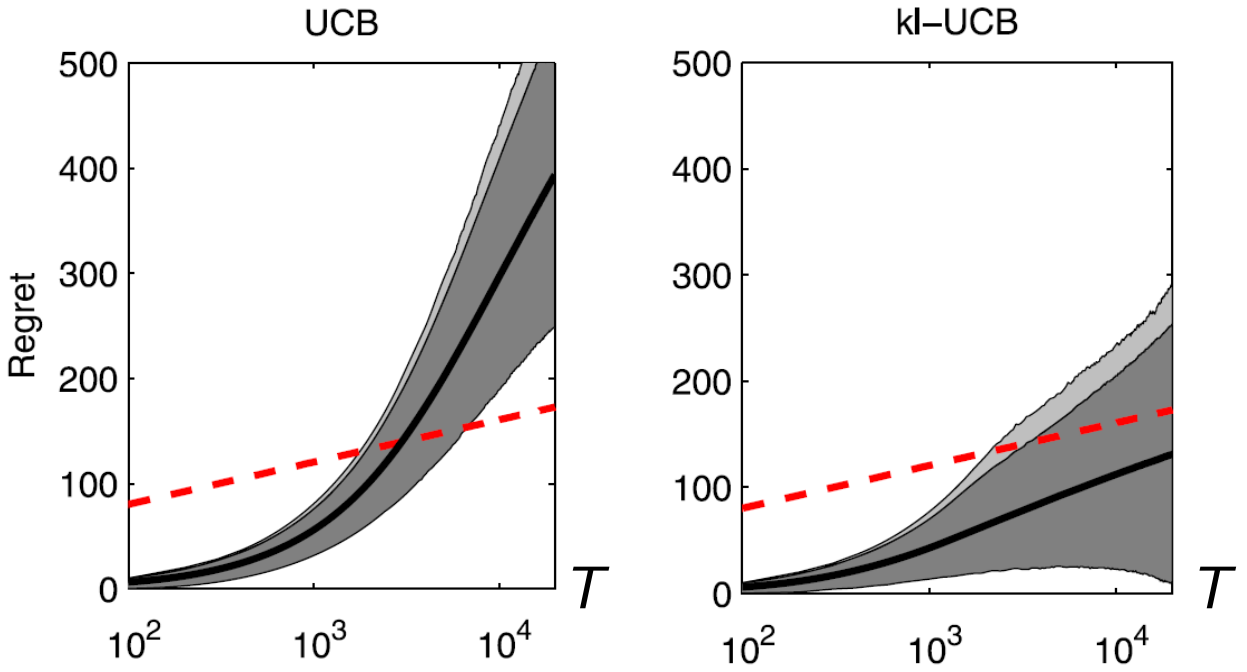
  - Algorithm based on this score is called *KL-UCB*

Improvement of UCB Algorithm (2/2)

- Regret bound of KL-UCB [Burnetas+, 1996][Cappe+, 2013]:

$$\mathrm{E}[\mathrm{Regret}(T)] \leq \sum_{i \neq i^*} \frac{\Delta_i \log T}{D(\mu_i \| \mu^*)} + o(\log T)$$

  - Optimal up to the coefficient of the logarithmic term



[Cappe+, 2013]

Worst-case Scenario

- Regret bound of UCB algorithm:

$$\mathrm{E}[\mathrm{Regret}(T)] \leq \sum_{i \neq i^*} \frac{\log T}{2\Delta_i} + o(\log T)$$

  - Becomes meaningless for $\Delta \to 0$
  - Called a problem-dependent bound in this sense
- Problem-independent (or worst-case) regret bound of UCB: for all $T > K$ and some universal constant $c > 0$:

$$\mathrm{E}[\mathrm{Regret}(T)] \leq c\sqrt{KT \log T}$$

  - Regret increases sublinearly in K
  - There are also many studies on this scenario

Outline

- Classification of bandit problems
- Regret minimization for stochastic bandits
  - ε-greedy
  - UCB algorithm
  - Thompson sampling
- Algorithms for other formulations

Thompson Sampling (TS)

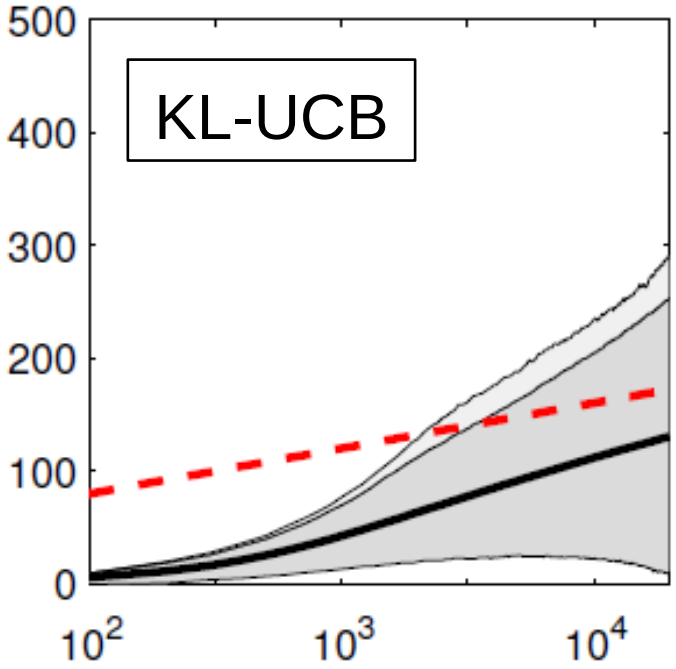
- A kind of Bayesian algorithm named after Thompson (1933)
- Pulls an arm randomly according to the posterior probability that the arm is the best

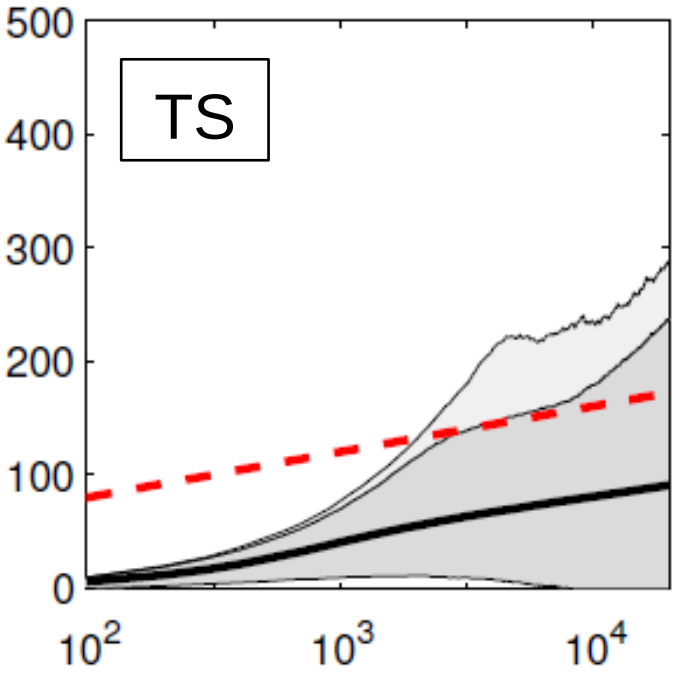
- Applicable to many complex settings (linear, contextual,...)
- Empirically performs well for many models
- (Usually) not sensitive to the choice of priors

- Asymptotic optimality is recently proved for some models
  - Bernoulli [Kaufmann+, 2012]
  - One-parameter exponential families [Korda+, 2013]
  - Gaussian with unknown variances [Honda+, 2014]

Performance of TS

- Often performs better than UCB
- Example: 10 ads, Click-Through-Rate:
  10%×1 arm, 5%×3 arms, 2%×3 arms, 1%×3 arms





[Kaufmann et al., 2012]

Advantage of Randomization

- TS pulls an arm randomly according to the posterior prob. that the arm is optimal
  - This posterior probability is hard to compute:

$$\pi(\mu_i \geq \max_{j \neq i} \mu_j | \{\boldsymbol{x}_i\})$$

$$= \int_0^1 \pi(\theta_i | \boldsymbol{x}_i) \left(\prod_{j \neq i} \int_0^{\theta_i} \pi(\theta_j | \boldsymbol{x}_j) \mathrm{d}\theta_j \right) \mathrm{d}\theta_i$$

prob. of $\mu_i = \theta_i$ prob. of $\mu_j \leq \theta_i$ for the other arms

  - Actually there is no need to compute this probability:
    1. Sample θ_i from posterior $\pi(\mu_i | \boldsymbol{x}_i)$ and
    2. Choose an arm maximizing θ_i .

Implementation of TS

- It is hard to compute posterior probability that each arm is optimal
- Suffices to simply sample θ_i from posterior $\pi(\mu_i|\boldsymbol{x}_i)$ and choose an arm maximizing θ_i.
  - Ex: Bernoulli model, uniform prior

Beta distribution

	arm 1	arm 2	arm 3
rewards	1,0,1,1	0,0	1,0,1
posterior for μ_i	$\mathrm{Be}(4,2)$	$\mathrm{Be}(1,3)$	$\mathrm{Be}(3,2)$
sample from the posterior			

Implementation of TS

- It is hard to compute posterior probability that each arm is optimal
- Suffices to simply sample θ_i from posterior $\pi(\mu_i|\boldsymbol{x}_i)$ and choose an arm maximizing θ_i.
  - Ex: Bernoulli model, uniform prior

	arm 1	arm 2	arm 3
rewards	1,0,1,1	0,0	1,0,1
posterior for μ_i	$\mathrm{Be}(4,2)$	$\mathrm{Be}(1,3)$	$\mathrm{Be}(3,2)$
sample from the posterior	0.78	0.16	0.68

Implementation of TS

- It is hard to compute posterior probability that each arm is optimal
- Suffices to simply sample θ_i from posterior $\pi(\mu_i|\boldsymbol{x}_i)$ and choose an arm maximizing θ_i.
  - Ex: Bernoulli model, uniform prior

	arm 1	arm 2	arm 3
rewards	1,0,1,1,0	0,0	1,0,1
posterior for μ_i	$\mathrm{Be}(4,3)$	$\mathrm{Be}(1,3)$	$\mathrm{Be}(3,2)$
sample from the posterior			

Implementation of TS

- It is hard to compute posterior probability that each arm is optimal
- Suffices to simply sample θ_i from posterior $\pi(\mu_i|\boldsymbol{x}_i)$ and choose an arm maximizing θ_i.
  - Ex: Bernoulli model, uniform prior

	arm 1	arm 2	arm 3
rewards	1,0,1,1,0	0,0	1,0,1
posterior for μ_i	$\mathrm{Be}(4,3)$	$\mathrm{Be}(1,3)$	$\mathrm{Be}(3,2)$
sample from the posterior	0.67	0.19	0.81

Implementation of TS

- It is hard to compute posterior probability that each arm is optimal
- Suffices to simply sample θ_i from posterior $\pi(\mu_i|\boldsymbol{x}_i)$ and choose an arm maximizing θ_i.
  - Ex: Bernoulli model, uniform prior

	arm 1	arm 2	arm 3
rewards	1,0,1,1,0	0,0	1,0,1,1
posterior for μ_i	$\mathrm{Be}(4,3)$	$\mathrm{Be}(1,3)$	$\mathrm{Be}(4,2)$
sample from the posterior			

Implementation of TS

- It is hard to compute posterior probability that each arm is optimal
- Suffices to simply sample θ_i from posterior $\pi(\mu_i|\boldsymbol{x}_i)$ and choose an arm maximizing θ_i.
  - Ex: Bernoulli model, uniform prior

	arm 1	arm 2	arm 3
rewards	1,0,1,1,0	0,0	1,0,1,1
posterior for μ_i	$\mathrm{Be}(4,3)$	$\mathrm{Be}(1,3)$	$\mathrm{Be}(4,2)$
sample from the posterior	0.58	0.61	0.77

Relation between TS and UCB

Thompson sampling:

- (prob. that arm i is pulled)
  = (posterior prob. p_i that the arm i is the best)
- Waits for $1/p_i$ rounds in average to be pulled next time

$\approx$ Pulls an arm i if $t > 1/p_i \Leftrightarrow p_i > 1/t$

UCB algorithm:

- Pulls an arm based on the upper bound of the expectation with significance level $1/t$

- TS does not use approximation by virtue of randomization and can avoid redundant exploration

Comparison of Algorithms for Regret Minimization

- ε-greedy:
  - Simple but cannot always achieve a logarithmic regret
- UCB:
  - Always achieves a logarithmic regret
  - Often used for many extensions of bandit problems to give a theoretical guarantee
- Thompson sampling:
  - Often perform better experimentally
  - Theoretical guarantee is difficult (and breaks down for some complex settings)

Outline

- Classification of bandit problems
- Regret minimization for stochastic bandits
- Algorithms for other formulations
  - Best Arm Identification (BAI)
  - Adversarial bandits
  - Nonstationary bandits
  - Linear / contextual bandits
  - Dueling bandits

Stochastic Bandits

- Pulls an arm at each round *t*
- Reward from the arm *i* follows Bernoulli distribution Ber(μ_i):
- Can observe the reward only for the pulled arm

Best arm identification:

- Fixed confidence: detect the best arm through rounds as few as possible with success prob. $\geq 1 - \delta$
- Fixed budget: detect the best arm after *T* rounds with success probability as high as possible
  - Many problems are left open in the fixed budget setting

Difference from Regret Minimization

- Algorithms for regret minimization pull seemingly suboptimal arms only for a logarithmic number of rounds
- Numbers of samples are biased too much between arms
- Ex: two arms, rewards follow normal distributions
  $N(\mu_1, \sigma^2)$, $N(\mu_2, \sigma^2)$ with known variance σ^2
  - Best arm identification = worst arm identification
  - Algorithms pulling arms in a biased way cannot be optimal
- The numbers of samples are the same order between arms under “good” BAI algorithms

Validity of Using UCB

- Main problem in regret minimization and BAI: whether seemingly the best arm $\hat{i}^*$ is really the best or another arm j has a larger expectation
- Regret minimization:
  - Seemingly the best arm is pulled very frequently
    - Confidence interval concentrates to the true value
  - Comparison between $UCB_{\hat{i}^*}$ and UCB_j

  $\approx$ Comparison between $\mu_{\hat{i}^*}$ and UCB_j

  $\approx$ Hypothesis testing of $\mu_{\hat{i}^*} \geq \mu_j$ with confidence level $1/t$
- BAI: numbers of samples are the same order between arms
  - The above argument does not apply anymore

Lower Confidence Bound (LCB)

- In BAI it is necessary to consider the possibility that seemingly the best arm $\hat{i}^*$ is overestimated:
  - $\mu_{\hat{i}^*}$ may be much smaller than its empirical estimate $\hat{\mu}_{\hat{i}^*}$
- Convenient to consider both of
  - Lower confidence bound for seemingly the best arm

$$\mathrm{LCB}_i = \hat{\mu}_i - \sqrt{\frac{\beta(N_i(t), \delta)}{2N_i(t)}}$$

  - Upper confidence bound for the other arms

$$\mathrm{UCB}_i = \hat{\mu}_i + \sqrt{\frac{\beta(N_i(t), \delta)}{2N_i(t)}}$$

LCB-based Algorithms

- LUCB [Kalyanakrishnan+, 2012] and UGapE [Gabillon+, 2012] algorithms for fixed confidence scenario:
  1. Compute the arm $\hat{i}_* = \mathrm{argmax}_{i \neq \hat{i}^*} \mathrm{UCB}_i$
  2. Pull both arms $\hat{i}^*$ and $\hat{i}_*$ (LUCB algorithm) or pull the arms $\hat{i}^*$ or $\hat{i}_*$ such that the number of samples is smaller (UGapE algorithm)
  3. If $\mathrm{LCB}_{\hat{i}^*} > \mathrm{UCB}_{\hat{i}_*}$ then output $\hat{i}^*$ as the best arm
- For $\beta(n, \delta) = \log \frac{5Kn^4}{4\delta}$ the algorithms outputs i^* with prob. $\geq 1 - \delta$ and the average number of samples is

$$O\left(\sum_{i \neq i^*} \frac{1}{\Delta_i^2} \log \frac{1}{\delta}\right)$$

Outline

- Classification of bandit problems
- Regret minimization for stochastic bandits
- Algorithms for other formulations
  - Best Arm Identification (BAI)
  - Adversarial bandits
  - Nonstationary bandits
  - Linear / contextual bandits
  - Dueling bandits

Adversarial Bandits

- Reward from each arm $x_{i,t} \in [0, 1]$ is determined by an adversary so that the player incurs loss

	t=1	t=2	t=3	t=4	Total
arm 1	0.8	0.5	0.2	0.3	1.8
arm 2	0.3	0.9	0.1	0.8	2.1
arm 3	0.4	0.1	0.4	0.6	1.5

- regret = maximum cumulative reward of a single arm
  - actual cumulative reward
  = 2.1 - 1.5 = 0.6
- No need to assume some reward distribution

Hedge Algorithm

- An algorithm for online learning [Freund+, 1997]
  - Setting where rewards from all arms are observable
- Parameter: $\eta > 0$
- Initialization: $w_1, w_2, \ldots, w_K := 1/K$
- Loop:
  1. Pull an arm according to probability $p_i = \frac{w_i}{\sum_{i'} w_{i'}}$
  2. Update each weight by $w_i := w_i e^{\eta x_i}$
- Arms with large rewards have large weight
- Regret is $O(\sqrt{T \log K})$ for appropriately chosen $\eta > 0$

EXP3 Algorithm

- Bandit setting: observes reward only for the pulled arm
- Idea: replace the reward with its unbiased estimator:

$$\hat{X}_i := \begin{cases} X_i / p_i & i \text{ is pulled}, \\ 0 & \text{otherwise}, \end{cases} \quad \mathrm{E}[\hat{X}_i] = p_i \mathrm{E}[X_i / p_i] = \mathrm{E}[X_i]$$

- Exponential-weight policy for exploration and exploitation:
  1. Pull an arm with prob. $p_i = (1-\gamma)\frac{w_i}{\sum_{i'} w_i} + \frac{\gamma}{K}$
  2. Update the weight by $w_i := w_i e^{\eta \hat{x}_i}$
  - Regret is $O(\sqrt{TK \log K})$ for appropriately chosen γ, η

[Auer+, 2002]

Outline

- Classification of bandit problems
- Regret minimization for stochastic bandits
- Algorithms for other formulations
  - Best Arm Identification (BAI)
  - Adversarial bandits
  - Nonstationary bandits
  - Linear / contextual bandits
  - Dueling bandits

Nonstationary Bandits

- Reward distribution may change non-stationarily
  (e.g, news recommendation)
- Regret for stationary setting:

$$\text{regret}(T) = \sum_{t=1}^{T} \left(\max_{i \in \{1,\cdots,K\}} \{\mu_i\} - \mu_{i_t} \right)$$

- Regret for nonstationary setting:

$$\text{regret}(T) = \sum_{t=1}^{T} \left(\max_{i \in \{1,\cdots,K\}} \{\mu_{i,t}\} - \mu_{i_t,t} \right)$$

  - Regret is at least $O(\sqrt{T})$ in the problem-dependent scenario if the distribution changes once [Garivier+, 2011]

Discounted UCB Algorithm

- The original UCB: $\text{UCB}_i = \hat{\mu}_i + \sqrt{\frac{\xi \log t}{N_i(t)}}$
- Discounted UCB: $\text{UCB}_i = \frac{S_i(t)}{W_i(t)} + \sqrt{\frac{\xi \sum_{j=1}^{K} W_j(t)}{W_i(t)}}$

$$S_i(t) = \sum_{t' \in \{1,\cdots,t\}} 1[i_{t'} = i]\gamma^{t-t'} X_i(t')$$

$$W_i(t) = \sum_{t' \in \{1,\cdots,t\}} 1[i_{t'} = i]\gamma^{t-t'}$$

  - Gives weight $\gamma^i \leq 1$ for the observation i rounds before
  - Becomes UCB when $\gamma = 1$

Regret of Discounted UCB

- Discounted UCB: $\mathrm{UCB}_i = \frac{S_i(t)}{W_i(t)} + \sqrt{\frac{\xi \sum_{j=1}^{K} W_j(t)}{W_i(t)}}$

$$S_i(t) = \sum_{t' \in \{1,\cdots,t\}} 1[i_{t'} = i] \gamma^{t-t'} X_i(t')$$

$$W_i(t) = \sum_{t' \in \{1,\cdots,t\}} 1[i_{t'} = i] \gamma^{t-t'}$$

  - Achieves problem-dependent regret $O(\sqrt{MT} \log T)$ for $\xi > 2$ and $\gamma = 1 - O(M/T)$
    - M: number of times that the distribution changes

Outline

- Classification of bandit problems
- Regret minimization for stochastic bandits
- Algorithms for other formulations
  - Best Arm Identification (BAI)
  - Adversarial bandits
  - Nonstationary bandits
  - Linear / contextual bandits
  - Dueling bandits

Linear Bandits

- Each arm (often called an *action* in this setting) i is represented by a feature vector $a_i \in D \subset \mathbb{R}^d$
- Reward: $x_i(t) = \theta^\top a_i + \epsilon_i$ (θ : unknown, $\mathrm{E}[\epsilon_i] = 0$)
- Ex: design of website:

$$a_i = \begin{pmatrix} \text{size of ad} \\ \text{size of font} \\ \text{layout} \\ \vdots \end{pmatrix}$$

Contextual Bandits

- Context depending on time is given at each round before choosing the actions
  - Weekday or holiday
  - Location of the user
  - Access history of the user
  - Expert advice
- Usually formulated as the linear bandit where feature depends on time: $x_i(t) = \theta^\top a_{i,t} + \epsilon_i$
  - The best action may depend on t

Algorithms for Linear Bandits

- Naive idea: ignore the feature and regard each action $a_i \in D$ as an independent arm
  - Suffers O(|D|) regret, often exponential in d
  - If D is a continuous set then inapplicable
- Regret can be bounded linearly in d by a UCB-type algorithm called LinUCB [Dani+, 2008]

TS for Linear Bandits

- Assume θ has prior distribution $N(0, \sigma_0^2 I)$ (or others)
- If the noise is a Gaussian $\epsilon_i \sim N(0, \sigma^2)$ then the posterior of θ is $\pi(\theta | A_t, X_t) = N(B_t^{-1} X_t,\ \sigma^2 B_t^{-1})$, where
  - $A_t = (a_{i_1,1}, a_{i_2,2}, \cdots, a_{i_t,t})$: chosen features
  - $X_t = (x_{i_1}(1), x_{i_2}(2), \cdots, x_{i_t}(t))$: observed rewards
  - $B_t = \dfrac{\sigma^2}{\sigma_0^2} + A_t^\top A_t$
- Thompson sampling for linear (or contextual) bandits:
  1. Randomly generate θ according to $N(B_t^{-1} X_t,\ \sigma^2 B_t^{-1})$
  2. Choose the action i maximizing $\theta^\top a_{i,t}$

TS for Online Advertisement

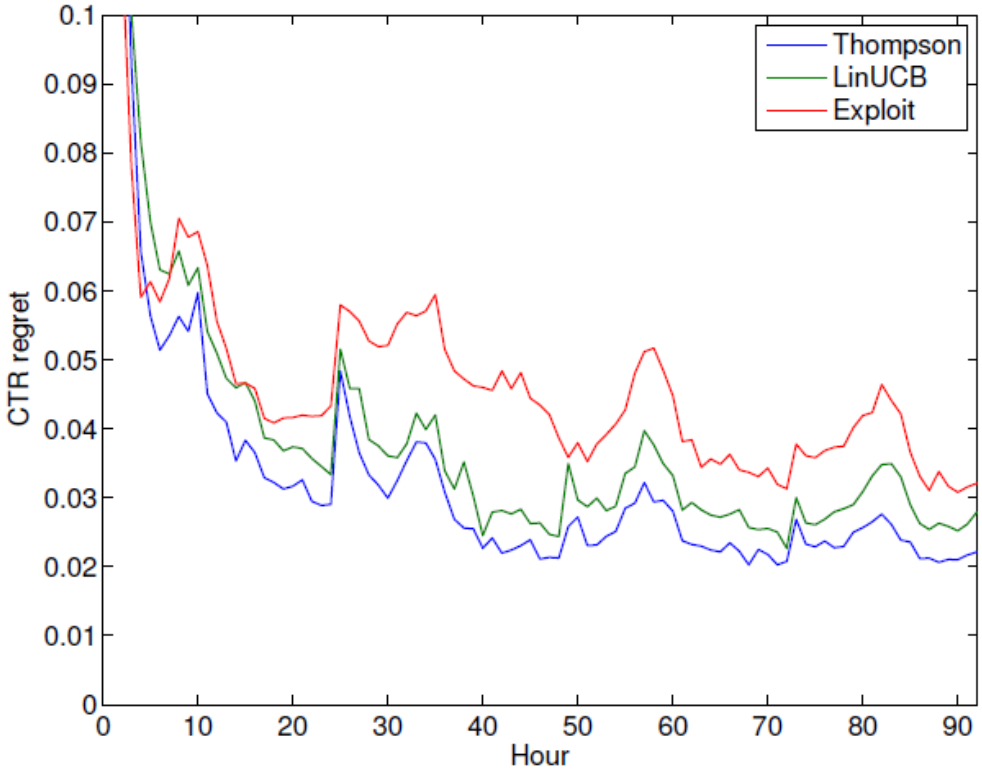
- Contextual bandit
  (user info,
  access history)

- Comparison of CTR
  (Click Through Rate)

- Logistic regression model:

$$x_i(t) \sim \text{Ber}\left(\frac{e^{\theta^\top a_{i,t}}}{1+e^{\theta^\top a_{i,t}}}\right)$$

(can be implemented similarly)



[Chapelle & Li, 2011]

Outline

- Classification of bandit problems
- Regret minimization for stochastic bandits
- Algorithms for other formulations
  - Best Arm Identification (BAI)
  - Adversarial bandits
  - Nonstationary bandits
  - Linear / contextual bandits
  - Dueling bandits

Dueling Bandits

- A formulation mainly for human preferences
  (e.g. music, movies...)
  - “Reward” of each arm is not explicitly observable
  - Relative comparison is still possible
- Ex: interleaved filtering
  - Optimization of website search engines
  - Choose a pair (i_1, i_2) of search algorithms for each query
  - Return the list of websites where outputs from two algorithms are interleaved
  - Regard “i_1 was preferred to i_2” if a result from i_1 is clicked
  - Regard “i_2 was preferred to i_1” otherwise

Formulation of Dueling Bandits

- K arms (search algorithms, music, movies,...)
- Chooses a pair of arms ($i_1(t)$, $i_2(t)$) at each round t
- Observes which arm is preferred, where

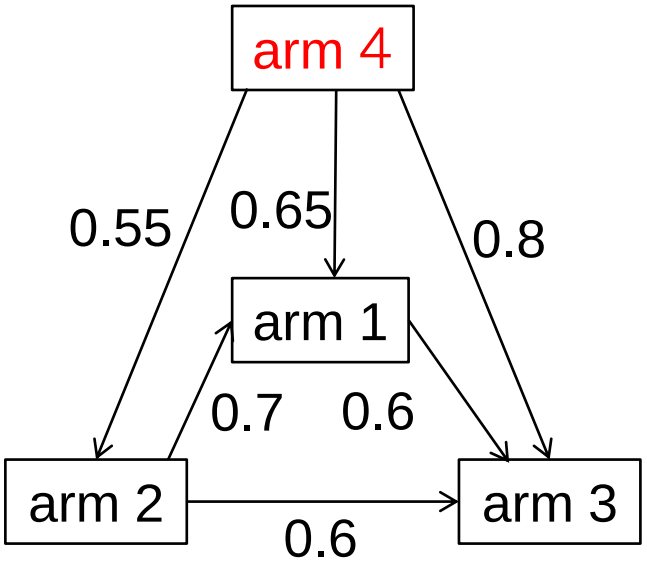
$$\Pr[i \text{ is preferred to } j] = \mu_{ij} = 1 - \mu_{ji}$$

  - $M = \{\mu_{ij}\}$ is called a preference matrix

- Arbitrariness in the definition of “reward” and “best arm”
  - Assumption of a total order
  - Assumption of Condorcet winner
  - Copeland winner
  - Borda winner

Assumption of Total Order

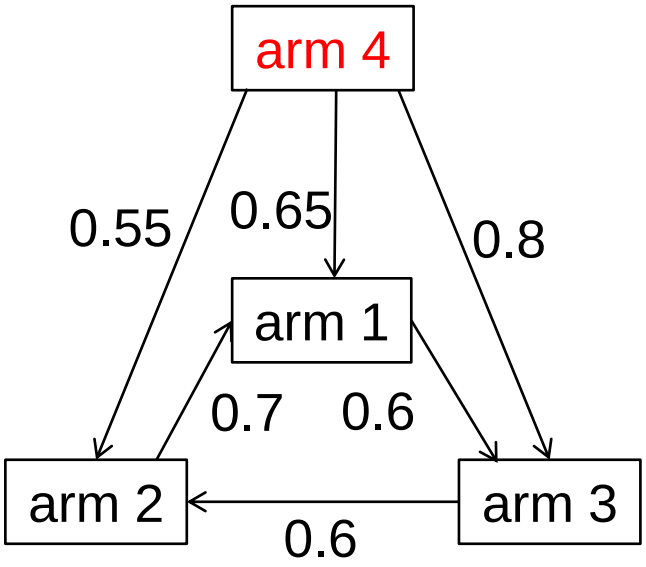
- Assumes a total order on the frequent winner



- Total order $4 \succ 2 \succ 1 \succ 3$ exists in this setting
- Natural to define $i^* = 4$ as the best arm
- Natural to set $\mu_{i^*,i} + \mu_{i^*,j} - 1 \geq 0$ as the loss for pair (i, j)

Assumption of Condorcet Winner

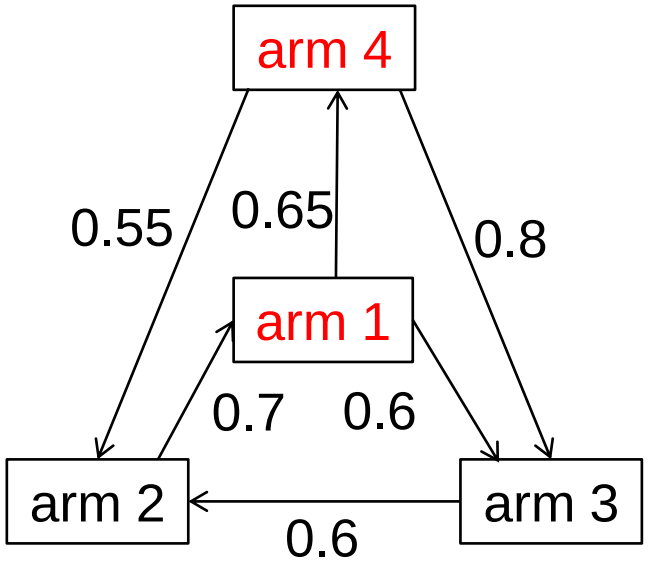
- Assume existence of an arm (Condorcet winner) that wins over .500 against all the other arms
  - Allows cyclic preference between the other arms



- Strictly weaker than the total order assumption, often exists
- Natural to set the loss $\mu_{i^*,i} + \mu_{i^*,j} - 1 \geq 0$ to for pair (i, j)

Copeland Winner

- Generalization of Condorcet winner (always exists)
- Arm *i* minimizing the number $L_i = \left|\{j : \mu_{ij} < 1/2\}\right|$ of the other arms beating the arm *i*



  - Natural to set the loss by $L_i + L_j - 2L_{i^*} \geq 0$ for pair (*i*, *j*)

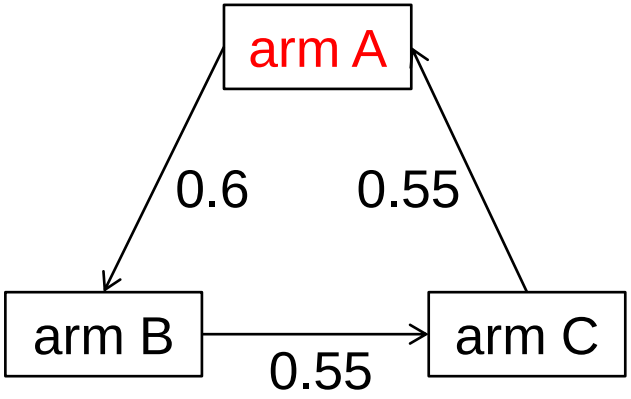
Borda Winner

- Another definition of the winner that always exists
- Implicitly assumes that each user has ranking for K arms
  - There can be $K!$ rankings
  - Loss for the user is i when i-th favorite arm is presented
- When the fraction of user $r \in \{1, 2, \cdots, K!\}$ is p_r then Borda winner is the arm minimizing the average loss:

$$i^* = \mathrm{argmin}_i \sum_{r=1}^{K!} (\text{rank of } i \text{ for type-}r \text{ user}) \times p_r$$

- Borda winner equals the arm maximizing the winning rate over an arm chosen uniformly at random

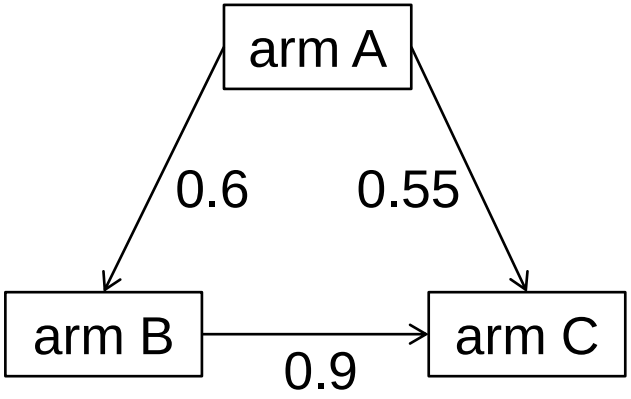
Borda Winner



- Natural in the sense that utility of users are considered
- Not compatible with Condorcet winner

$A \succ B \succ C$	$A \succ C \succ B$	$B \succ A \succ C$	$B \succ C \succ A$	$C \succ A \succ B$	$C \succ B \succ A$
0.2	0.15	0.1	0.25	0.25	0.05

Borda Winner



- Natural in the sense that utility of users are considered (but utilities other than the linear utility cannot be considered)
- Not compatible with Condorcet winner
  - Arm A is the Condorcet winner but
    Arm B is the Borda winner in this example

Algorithm for Dueling Bandits

- The original bandit problem:
  - UCB: estimates an upper bound of the expectation of each arm with confidence level 1/*t*
  - TS: pulls an arm that is the best with posterior prob. larger than 1/*t*
  - MED (Minimum Empirical Divergence): pulls an arm that is the best with likelihood larger than 1/*t*
- The idea of MED can easily be applied to dueling bandits with Condorcet winner assumption: RMED (Relative MED)

[Komiyama+, 2013]

Likelihood for Condorcet Winner

- Condorcet winner: beats all the other arms over .500
- Consider an arm *i* currently beaten by $j \in \{1, \cdots, K\}$ with empirical winning rate $\hat{\mu}_{ij} < 1/2$
  - If *i* is the Condorcet winner then:
    - Arm *i* beats *j* with prob $\mu_{ij} \geq 1/2$
    - Winning rate $\hat{\mu}_{ij}$ appears with prob. $\leq \mathrm{e}^{-n_{ij} D(\hat{\mu}_{ij} \| 1/2)}$ (Chernoff-Hoeffding's inequality)
  - The arm *i* is the Condorcet winner with likelihood

$$L_i = \prod_{j:\hat{\mu}_{ij}<1/2} \mathrm{e}^{-n_{ij} D(\hat{\mu}_{ij} \| 1/2)}$$

- $\hat{i}^* := \mathrm{argmax}_i L_i$: most likely to be the Condorcet winner

RMED Algorithm

- MED algorithm:
  pulls an arm that is the best with likelihood larger than $1/t$

- RMED algorithm:

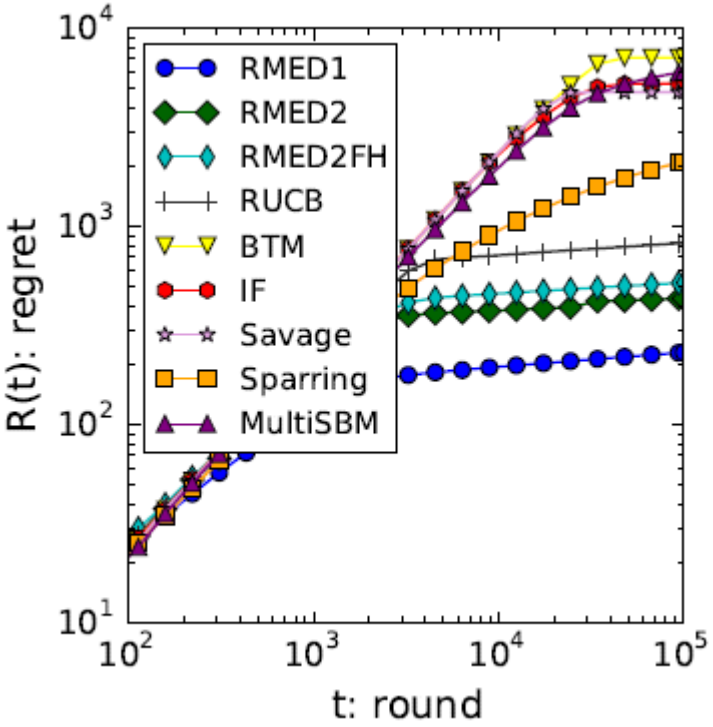
1. Compute L_i for each arm and $\hat{i}^* := \mathrm{argmax}_i L_i$
2. If there is an arm $i \neq \hat{i}^*$ such that $L_i > 1/t$ then:
   compare this arm i with arm $\mathrm{argmin}_{j \in J_i} \hat{\mu}_{ij}$
   and otherwise:
   compare pair $(\hat{i}^*, \hat{i}^*)$

Performance of RMED Algorithm

- RMED achieves regret

$$\mathrm{E}[\mathrm{regret}(T)] \leq \sum_{i \neq i^*} \frac{(\mu_{i^*,i} - 1/2)\log T}{d(\mu_{i^*,i} \| 1/2)} + o(\log t)$$

  - Optimal for most instances
- Simulation:
  - Dataset:
    preference of 16 types of Sushi
    [Kamishima, 2003]
  - Condorcet winner:
    mildly-fatty tuna (chu-toro)



[Komiyama+, 2013]

Outline

- Classification of bandit problems
- Regret minimization for stochastic bandits
- Algorithms for other formulations
  - Best Arm Identification (BAI)
  - Adversarial bandits
  - Nonstationary bandits
  - Linear / contextual bandits
  - Dueling bandits

Homework

- Choose one from the following two assignments.

Assignment 1

- Choose and implement one (or more) algorithm(s) from UCB, KL-UCB and Thompson sampling for stochastic bandits with rewards following Bernoulli distributions and plot the behavior of average regrets.
  - Use the same parameters K *and* μ_i as p.35:
    - K=10 ads, Click-Through-Rate μ_i :
      10%×1 arm, 5%×3 arms, 2%×3 arms, 1%×3 arms
  - Run until 10,000-th rounds in one trial and repeat at least 100 independent trials to take the average

Assignment 2

- Choose one (or more) example(s) of real-world problems where a bandit algorithm will be applicable and discuss the following points
  (both examples in the lecture and others are OK).

1. Discuss what formulation (stochastic / adversarial, regret minimization / best arm identification,...) of bandit problems is close to the example.
2. Give some difference of the setting between the ideal formulation in the lecture and the actual problem (and, if possible, discuss what kind of modification for algorithms will improve the performance).